Joint Longitudinal Viewer (JLV) 2.10

Veterans Affairs Enterprise Cloud

Deployment, Installation, Backout, and Rollback Guide (DIBRG)

November 2021

Version 1.0

Department of Veterans Affairs

Office of Information and Technology (OIT)

Revision History

Date	Version	Description	Author
04/28/2021	0.1	Initial creation of document from last approved	Liberty ITS
7/13/2021	0.2	Edit for VAEC information	Liberty ITS
08092021	0.3	Additional AWS information	Liberty ITS
08/13/2021	0.4	Additional AWS information	Liberty ITS
08/24/2021	0.5	Additional AWS information	Liberty ITS
08/31/2021	0.6	Updated per corrections from adjudication meeting	Liberty ITS
09/22/2021	0.7	Review	LeAnn Miller
09/29/2021	0.8	Additional AWS information	Liberty ITS
10/12/2021	0.9	Updated per corrections from adjudication meeting	Liberty ITS
11/09/21	1.0	Update to describe manual deployment	Liberty ITS

Artifact Rationale

This document describes the Deployment, Installation, Backout, and Rollback Guide (DIBRG) for Joint Longitudinal Viewer (JLV) releases going into the Department of Veterans Affairs (VA) Enterprise Cloud (VAEC). The Guide includes information about system support, issue tracking, and escalation processes, and it identifies the roles and responsibilities involved in all those activities. Its purpose is to provide clients, stakeholders, and support personnel with a smooth transition to the new product or software, and this artifact should be structured appropriately to reflect the particulars of these procedures at a single or at multiple locations.

Per the Veteran-focused Integrated Process (VIP) Guide, the Deployment, Installation, Backout, and Rollback Guide is required to be completed prior to Critical Decision Point 2 (CD2), with the expectation that it is updated throughout the life cycle of the project for each build as needed.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

Born from a joint Department of Defense (DoD)–Department of Veterans Affairs (VA) venture called JANUS, Joint Longitudinal Viewer (JLV) was directed by the Secretary of the VA and the Secretary of Defense in early 2013 to further support interoperability between the two departments. JLV is a centrally hosted, Java-based web application managed as two similar but distinct products - one tailored for DoD use and another tailored for VA use. Each JLV product is deployed to its respective DoD and VA hosting environments. Although separately hosted, the respective applications use several shared data services. The browser-based, Graphical User Interface (GUI) provides an integrated, read-only view of Electronic Health Record (EHR) data from VA, DoD, and community partners within a single application.

JLV eliminates the need for VA and DoD clinicians to access disparate viewers. The GUI retrieves clinical data from several native data sources and systems, then presents it to the user via widgets, each corresponding to a clinical data domain.

Users can create and personalize tabs, drag and drop widgets onto tabs, sort data within a widget's columns, set date filters, and expand a widget for a detailed view of patient information. Within each widget, a blue circle indicates VA data; an orange square indicates DoD data; a purple hexagon indicates community partner data; and a green triangle indicates Cerner Millennium Federal Electronic Health Record (FEHR) data.

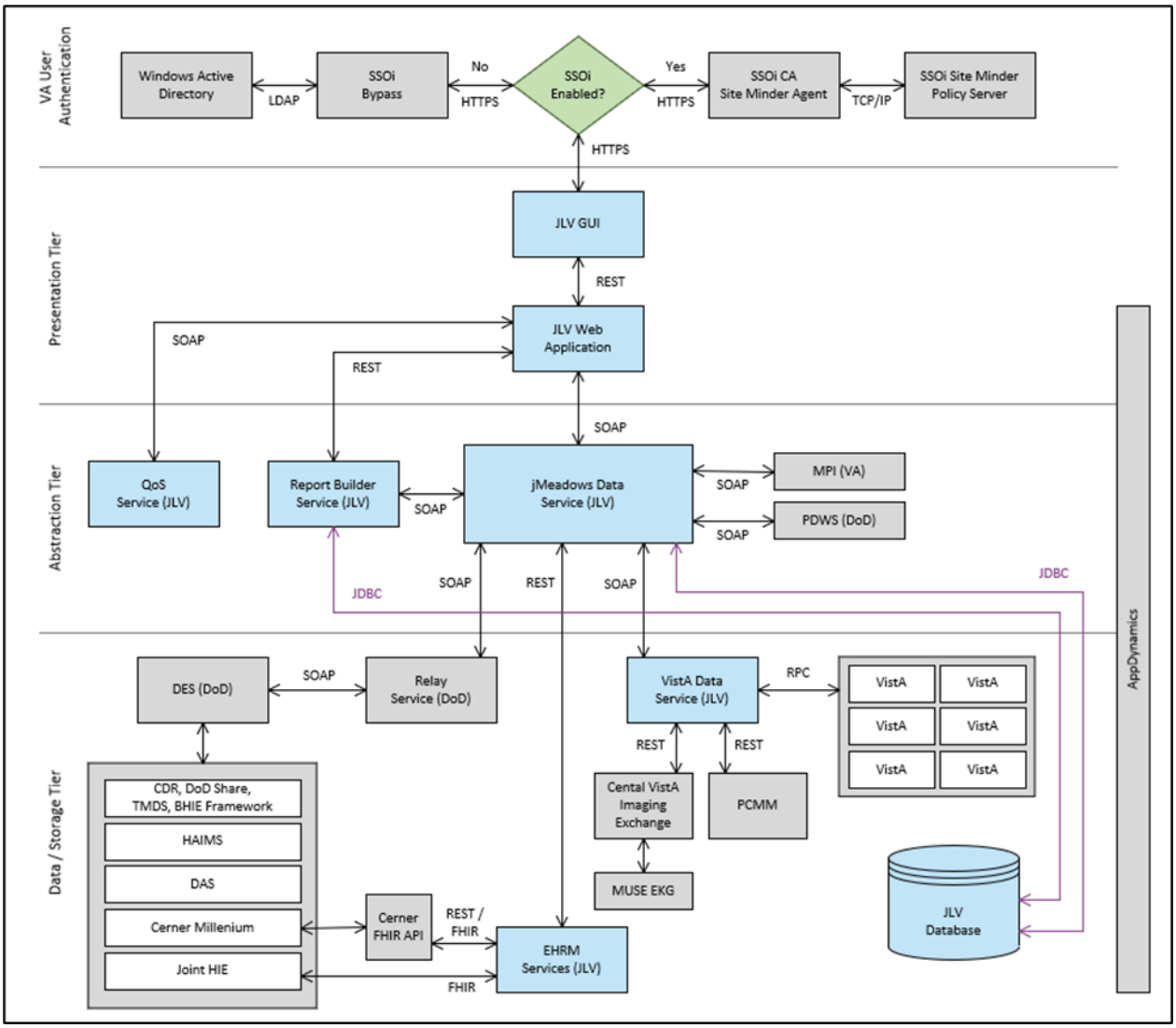
1.1. Purpose

The Deployment, Installation, Backout, and Rollback Guide (DIBRG) provides a single, common document that defines the ordered, technical steps required to install and deploy the JLV product. Further, it outlines the steps to back out of the installation and roll back to the previously installed version of the product if necessary. The installation process is completed in the VA Enterprise Cloud (VAEC).

System design specifications and diagrams can be found in the REDACTED

Figure 1 illustrates the three tiers (presentation, abstraction, and data/storage) of the JLV architecture, the location of JLV system components within the tiers, the external systems that JLV communicates with in Enterprise environments, and the communication protocols and authentication methods used. In the diagram, components of JLV are shaded blue; external systems are shaded grey.

Figure 1: JLV Architecture and Components[1]



[1] Active Directory (AD), Application Program Interface (API), Bidirectional Health Information Exchange (BHIE), Central VistA Imaging Exchange (CVIX), Clinical Data Repository (CDR), Computer Associates (CA), Data Access Service (DAS), Data Exchange Service (DES), Electronic Health Record Modernization (EHRM), Fast Healthcare Interoperability Resources (FHIR), Healthcare Artifact and Image Management Solution (HAIMS), Health Information Exchange (HIE) HyperText Transfer Protocol Secure (HTTPS), Internet Protocol (IP), Java Database (DB) Connectivity (JDBC), Lightweight Directory Access Protocol (LDAP), Master Person Index (MPI), Military Health System (MHS), Patient Discovery Web Service (PDWS), Quality of Service (QoS), Remote Procedure Calls (RPCs), Representational State Transfer (REST), Single Sign On Internal (SSOi), Simple Object Access Protocol (SOAP), Theater Medical Data Store (TMDS), Transmission Control Protocol (TCP), Veterans Health Information Systems and Technology Architecture (VistA) Data Service (VDS)

1.2. Dependencies

JLV is dependent on ancillary systems that connect the application to specific data sources. If any of these sources encounter a disruption in data services, the data is not pulled into JLV.

JLV is also dependent on internal VA update requirements, including DB updates, machine image updates, and security patches. If any of the VA Enterprise operational procedures disrupt the normal operation of JLV, the application is not fully functional.

The physical environment is Amazon Web Services (AWS) - GovCloud western region. The VAEC provides security and environmental control over the JLV servers and are restricted by Elevated Privilege (EP) access. Project personnel request EP access by submitting the *JLV Linux (Centrify) and Windows Access Requirements* spreadsheet to the VA project manager (PM) / Contracting Officer Representative (COR) for approval via the Elevated Permissions Access System. AWS Console access is requested through the VAEC ticketing system. Any delay in granting initial EP access hinders the ability to respond to technical impacts to the servers.

Developers create the code which describe each individual service, the code is saved as a template. Joint Longitudinal Viewer (JLV) in the VAEC uses Docker, which is an industry standard, open-source platform making it possible to create and run containers. Containers isolate application components and their dependencies. Containers also provide virtualization and scalability. Containers bundle together an application's code with all the runtime libraries, system tools and setting needed to support the application. A container is a logical environment created on a computer where an application can run.

Containers and virtual machines are both "packages." A container is a package that includes your application and everything it needs to run, aside from the operating system. A virtual machine is a package that includes your application and everything it needs to run, including the operating system.

The first step in the process is to create a template that describes all the resource properties and the configuration of these needed resources. This template is then built and stored in a repository which can be pulled and ran in all environments from development to production.

Container technology has allowed applications to be decomposed into different smaller services. This technological approach is named Microservices. Microservices are small independent services that communicate over well define Application Programming Interfaces (APIs). Services are built as independent components for business capabilities that run each application process as a service and performs a single function. Each component service in a microservices architecture can be developed and scaled without affecting the functioning of other services. Services do not need to share any of their code or implementation with other services. Any communication between individual components between individual components happens via well-defined APIs.

JLV in VAEC is built on a microservices architecture.

JLV VAEC is built using:

- AWS CodeBuild – continuous integration service that builds source code.
- Docker – containerization platform.
- Amazon Elastic Container Service (ECS) – container orchestration service.
- Elastic Container Registry (ECR) – Docker container registry that stores container images.
- AWS Lambda – serverless, event-driven service that runs code without provisioning hardware.
- Amazon S3 – scalable, reliable, and durable object storage.
- Elastic Load Balancing (ELB) – distributes incoming application traffic across multiple targets in one or more availability zone.
- Amazon Fargate – serverless compute engine that manages server infrastructure.
- Amazon CloudFormation – infrastructure as code.
- Amazon Elastic File Storage (EFS) – serverless file system that automatically shrinks and grows with no need for management or provisioning.
- Microsoft SQL (Structured Query Language) – relational database engine developed by Microsoft
- YAML (Yet Another Markup Language) – human friendly data serialization language for all programming languages.
- Java – object-oriented programming language.

Figure 2: Sample YAML Template

```yaml
AWSTemplateFormatVersion: 2010-09-09
Description: Production - ECR Repositories storing JLV docker images

Parameters:
  pEnvName:
    Type: String
    AllowedValues: [ silver, gold, pre-prod, prod ]
    Description: Environment Name - silver, gold, pre-prod, prod

Resources:
  rJLVRepo:
    Type: AWS::ECR::Repository
    Properties:
      ImageScanningConfiguration:
        ScanOnPush: true
      ImageTagMutability: MUTABLE
      RepositoryName: !Sub jlv-web-${pEnvName}
```

Developers create the code which describe each individual service and the code is saved as a template. These templates are converted into Docker images.

Docker is a technology that provides the tools to build, run, test, and deploy distributed applications that are based on Linux containers. Amazon ECS uses Docker images in task definitions to launch containers as part of tasks in clusters.

Table 1: Project Naming Convention

Project	Service	Environment
JLV	jMeadows	Production
JLV	JLV	Production
JLV	VistA Data Service (VDS)	Production
JLV	Report Builder (JLVRB)	Production
JLV	JLV QOS (Quality of Service)	Production
JLV	EHRM	Production

The JLV team has created one Cluster in Production – Cluster A. Our goal in the future to have two cluster by creating Cluster B for Blue/Green deployments. The designation of which cluster is Blue or Green will be determined by which is live at the time.

Until the Blue/Green architecture is fully implemented, we will rely on manual deployments. To manually deploy JLV, each service will have to be built in AWS CodeBuild which will push the new image to AWS Elastic Container Registry (ECR). To deploy the new code, the task needs to be manually stopped which will cause the service to recreate a task with the latest container image from ECR.

When an application is developed and deployed to an AWS ECS Fargate Cluster, having two separate environments—blue and green—increases availability and reduces risk. The green environment is the production environment that handles live traffic. The newest application version will be deployed and tested in the blue environment. After sufficient testing, we then "swap" the Application Load Balancer (ALB) targets between the two environments.

1.3. Constraints

Not applicable to JLV.

2. Roles and Responsibilities

Table 2 and Table 3 list the project and DIBRG roles and responsibilities.

Table 2: Project Roles

Name	Title/Group	Company
REDACTED	JLV Program Manager (PgM)	VA
REDACTED	VA JLV Tester	VA
REDACTED	JLV Project Manager (PM)	VA

REDACTED	JLV Test Lead	VA
REDACTED	JLV Functional Analyst	VA
REDACTED	Program Associate	Liberty ITS
REDACTED	Tester	Liberty ITS
REDACTED	Dev Ops Engineer	Liberty ITS
REDACTED	Java Developer	Liberty ITS
REDACTED	Windows Sys Admin	Liberty ITS
REDACTED	Application Architect (Project Support)	Liberty ITS
REDACTED	Ops Engineer	Liberty ITS
REDACTED	Product Owner (Project Support)	Liberty ITS
REDACTED	Contract Program Manager	Liberty ITS
REDACTED	Mid Java Developer	Liberty ITS
REDACTED	Developer	Liberty ITS
REDACTED	Java Developer	Liberty ITS
REDACTED	Business Analyst (Project Support)	Liberty ITS
REDACTED	Database Administrator	Liberty ITS
REDACTED	Program Associate	Liberty ITS
REDACTED	Info Assurance	Liberty ITS
REDACTED	Java Developer	Liberty ITS
REDACTED	Tester	Liberty ITS
REDACTED	Tester	Liberty ITS
REDACTED	Configuration Manager (Project Support)	Liberty ITS
REDACTED	Operations Engineer	Liberty ITS
REDACTED	Ops Engineer	Liberty ITS
REDACTED	Tester	Liberty ITS
REDACTED	System Administrator	Liberty ITS
REDACTED	Operations	Liberty ITS
REDACTED	Sr. Automation Tester/Architect	Liberty ITS
REDACTED	Project Manager	Liberty ITS
REDACTED	Scheduler (Project Support)	Liberty ITS
REDACTED	JLV/CV tester	Liberty ITS
REDACTED	Automation Tester	Liberty ITS
REDACTED	Project Manager	Liberty ITS
REDACTED	AWS Cloud Arch	Liberty ITS
REDACTED	Test Engineer	Liberty ITS
REDACTED	Tester	Liberty ITS
REDACTED	Program Associate	Liberty ITS
REDACTED	Database Administrator	Liberty ITS
REDACTED	Java Developer	Liberty ITS
REDACTED	Sr Scrum Master – SAFe (Project Support)	Liberty ITS

REDACTED	Tester	Liberty ITS
REDACTED	Scrum Master	Liberty ITS
REDACTED	Sr. Java Developer	Liberty ITS
REDACTED	Sr. Java Developer	Liberty ITS
REDACTED	JAVA Support	Liberty ITS
REDACTED	Info Assurance	Liberty ITS
REDACTED	Tester	Liberty ITS

Please note that references to JLV Support in the table below indicate Liberty Team Operations and Engineers. See Table 7 for additional details regarding the Phase/Role column of Table 3.

Table 3: Deployment, Installation, Backout, and Rollback Roles and Responsibilities

Team	Phase/Role	Tasks
Enterprise Project Management Office (EPMO)	Approval for Release to Production	Review the Release Readiness Report (RRR) with JLV PM and Health Product Support (HPS) for approval; review and approve the Service Now (SNOW) board entry.
JLV Support	Deployment Coordination	• Plan and schedule deployments (including orchestration with vendors) • Determine and document the roles and responsibilities of those involved in deployments • Test for operational readiness
JLV Support	Installation	• Schedule installations • Ensure that the Authority to Operate (ATO) and certificate authority security documentation is in place • Validate through facility Points of Contact (POCs) to ensure that Information Technology (IT) equipment has been accepted using the asset inventory processes • Coordinate training See Section 4.8.2 for the detailed installation process.
JLV Support	Backout	• Confirm the availability of backout instructions and backout strategy. • Identify the criteria that triggers a backout See Section 5 for the detailed backout process.
JLV Support	Rollback	• Confirm the availability of rollback instructions and rollback strategy. • Identify the criteria that triggers a rollback See Section 6 for the detailed rollback process.
JLV Support	Post-Deployment	Provide software and system support

3. Deployment

The JLV deployment workflow is as follows.

1. Once EPMO approval is complete, the JLV Support team schedules the deployment.

2. JLV Support completes a SNOW Change Order (CHG).
3. The DevOps team will create a Docker container image with the latest application code and configurations using AWS CodeBuild.
4. The container image is tagged and pushed to Elastic Container Registry (ECR).
5. The Elastic Container Service (ECS) Task Definition is updated with the latest ECR image version.
6. After the ECS Service is updated to deploy the latest Task Definition, the Service will create the respective containers using the task definition.
7. The new image will have been tested in the lower environment (Pre-Prod), however, smoke testing is conducted in Production during maintenance hours.
8. Once the deployment is complete, Production testing is verified by the JLV Support team; please see Access Requirements and Skills Needed for Installation for additional information.
9. If there are any issues, we can revert the change by updating the services with the previous task definition and image located in ECR.

3.1. Timeline

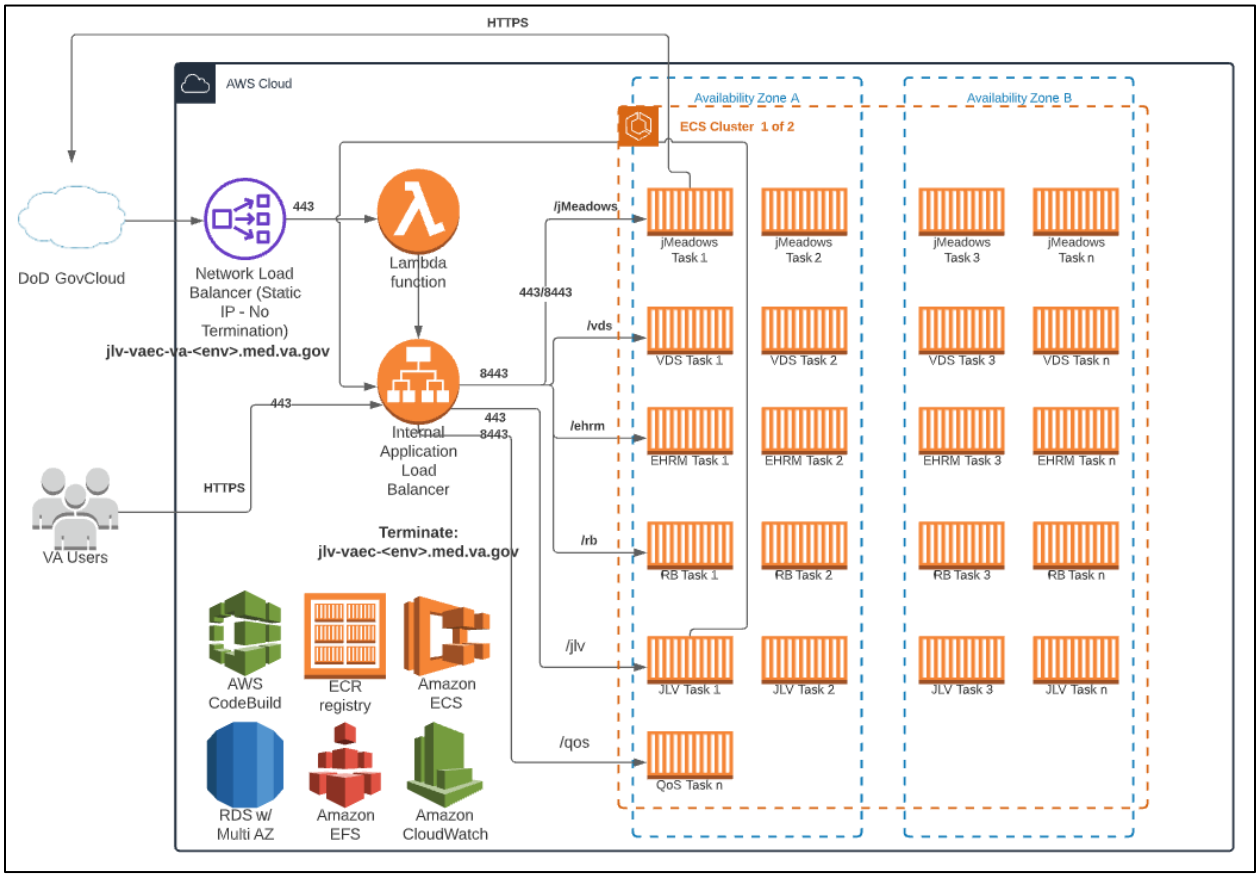
The deployment and installation have a duration of 8 hours.

3.2. Site Readiness Assessment

JLV is a Production, enterprise-wide application hosted in VAEC. All site readiness assessments are completed by JLV Operations.

3.2.1. Deployment Topology (Targeted Architecture)

Figure 3: JLV Topology



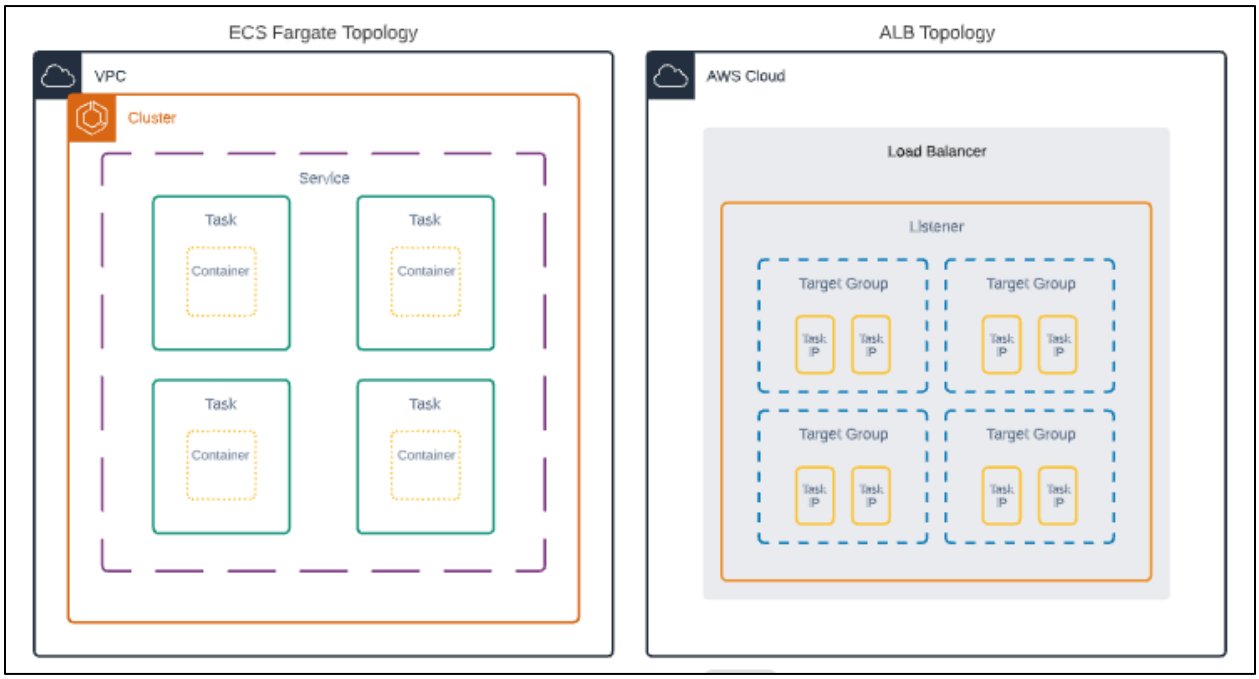
Note: For this release only one cluster will be available for the blue/green deployment. The second cluster is in progress and will be available for future deployments.

JLV is hosted on AWS with our application containerized using Docker images. The Docker images are built on AWS CodeBuild and pushed to Elastic Container Registry (ECR) image repository. The application is running in AWS Elastic Container Service (ECS) Fargate which pulls in new Docker images for each tier from ECR. Fargate abstracts the need to maintain virtual servers thus making it a serverless architecture. Each tier of JLV will function as an ECS which will orchestrate task/container placement and direct traffic from the Application Load Balancer (ALB). A Fargate Task runs one or more containers and gets its instructions from its Task Definition.

Each application runs on top of Tomcat 8.5 which connects to the SQL database (DB) hosted on Relational Database Server (RDS). The JLV web tier task runs two (2) containers, one for the SiteMinder Identity Access Management connection, and the other for the main application that uses Apache Httpd as a proxy to Tomcat. The Report Builder tier uses Amazon Elastic File Storage (EFS) to store report files. All tiers use Amazon CloudWatch for logging and monitoring. Logs are also shipped to the VAEC's Central Logging Solution to be viewed in Kibana.

There is a Network Load Balancer (NLB) setup for the DoD JLV connection which accepts traffic and forwards to the ALB. The resolution of the IP addresses of the ALB is maintained by a Lambda function because of the inability of ALBs to have static IPs.

Figure 4: JLV ECS Fargate and ALB Topology



3.2.2. Site Information (Locations, Deployment Recipients)

The host site for JLV is VAEC located in the AWS west region.

3.2.3. Site Preparation

All site preparation is completed by JLV Operations team.

JLV container images have the latest program updates and security patches. These updates are performed every time a new container image is built for a new version of the application.

Table 4 describes the preparation required by the site(s) prior to deployment.

Table 4: Site Preparation

Site	Problem/Change Needed	Features to Adapt/Modify to New Product	Actions/Steps	Owner
VAEC	Security Patches/Program Updates	Non-identifiable	Implement/Verify	JLV OPS

3.3. Resources

Descriptions of the hardware, software, facilities, and documentation are detailed in the following subsections.

3.3.1. Facility Specifics

VAEC is a cloud-based General Support System (GSS), hosted on Amazon Web Services GovCloud (AWS); FedRAMP package #: F1603047866. As per AWS's FedRAMP authorization, AWS is responsible for all Dynamic Routing Protocol (DRP) activities within this environment. AWS's DRP can be found in FedRAMP package #: F1603047866. The AWS is designed utilizing separate data centers (zones), and geographically separated regions. Currently AWS has two regions located in Iowa and Virginia.

The VA Enterprise Cloud AWS GovCloud High (VAEC) is a GSS that provides a secure application and hosting environment for VA applications, content, and utilities. These applications and services are used to deliver content to an audience made up of employees, veterans, contractors, partners across all VA medical centers and component facilities, Federal government, and the general public. Content and applications are provided by Veterans Benefits Administration (VBA), Veterans Health Administration (VHA), National Cemetery Administration (NCA), and VA-level support offices. VAEC provides the following services: Content delivery, Application Hosting and Management Services.

3.3.2. Hardware

The VAEC infrastructure is hosted by AWS GovCloud, a cloud service provider. The AWS GovCloud platform is used to provide a variety of hosting environments to suit a variety of needs. AWS GovCloud can support applications categorized up to "High" as rated in accordance with Federal Information Processing Standard (FIPS) 199. VA applications available to the public are hosted in AWS GovCloud.

A dedicated private data link (AWS Direct Connect) provides all connectivity for VA resources communicating to the environment. Virtual Private Clouds (VPCs) wrap the applications within AWS GovCloud to encapsulate network access. Access from the applications to VA internal resources such as Identity, Credential, and Access Management (ICAM) and Active Directory (AD) Services are conducted over the encrypted private data link to the VA Network.

VAEC is in two (2) regions with three (3) Availability Zones designed to allow United States (U.S.) government agencies, contractors, and customers to move sensitive workloads into the cloud for addressing specific regulatory and compliance requirements. AWS GovCloud does not manage logical access controls within the VAEC system boundary. VAEC offers the same level of security as other VA physical technology centers and supports existing VA security controls and certification requirements such as FISMA, Health Insurance Portability and Accountability Act of 1996 (HIPAA), HITECH, SAS-70, ISO 27001, FIPS 140-2 compliant end points, and PCI DS

Table 5 describes the hardware specifications required at each site prior to deployment. Please see Table 3 for details about the party or parties responsible for preparing the site to meet the hardware specifications.

Table 5: Virtual Machine (VM) Production Hardware Specifications (AWS Managed)

Required Hardware	Model	Configuration	Manufacturer	Number of Containers
ECS Fargate Task	Amazon Linux 2	vCPU 1 RAM 8 GB Storage 20 GB	Virtual	Automatically Scaled
JLV Web Container (main)	Amazon Linux 2	vCPU 1 RAM 16 GB (shared) Storage 20 GB	Virtual	Automatically Scaled
Single Sign on Internal (SSOi) Container (sidecar)	Amazon Linux 2	vCPU 1 RAM 16 GB (shared) Storage 20 GB	Virtual	Automatically Scaled
RDS Instance	SQL Server Enterprise 14.00.3049.1.v1	vCPU 16 RAM 64 GB Storage 500 GB	Virtual	Automatically Scaled

3.3.3. Software

Table 6 describes the software specifications required at each site prior to deployment. Please see Table 3 for details about the party or parties responsible for preparing the site to meet the software specifications.

Table 6: Software Specifications

Required Software	Make	Version	Manufacturer	Other
SQL Server Enterprise	N/A	14.00.3049.1.v1	Microsoft	N/A
Amazon Linux	N/A	2	Amazon	N/A
Apache	N/A	2.4	Apache	N/A
SiteMinder	N/A	12.51	CA Technologies	N/A
Tomcat	N/A	8.5	Apache	N/A

More information about SSOi server installation can be found in the *CA SiteMinder Apache Web Agent Install & Configuration Guide,* maintained by Identity and Access Management (IAM).

3.3.4. Communications

JLV Operations team performs JLV installation and deployment activities in the virtualized environments at VAEC utilizing the release-ready package. When possible, the installation is performed during off-hours to minimize the impact on users. We will use a a 60-minute window for downtime to manually deploy and test.

An overview of typical steps and/or communication during the implementation process is as follows:

1. The Configuration Manager submits a JLV release (RLSE) notification via SNOW/CHG Order
2. Plan the system upgrade and change notifications:
   a. Notify the JLV PgM and PM and the OIT PM/COR
3. Perform the installation/deployment:
4. Dev Ops coordinates with ITOPS to remove the current installation from service and deploy the new version into service
   a. VHA validates the installation of the new version
5. The Scrum Master notifies the stakeholders and Product team that systems are updated.

3.3.4.1. Deployment/Installation/Backout Checklist

Table 7 captures the coordination effort and documents the day/time/individual when each activity (deploy, install, back-out) is completed for a project.

Table 7: Deployment Installation and Backout Checklist

Activity	Day	Time	Completed By
Deployment	Decision by VA OIT	Deployment dependent on a planned maintenance ticket	JLV Operations
Installation	Deployments staged	Coordinated with JLV OPS and DEV OPS	JLV Operations
Backout	As needed	As needed, with a time estimate to be communicated to stakeholders when determined	JLV Operations

4. Installation

4.1. Preinstallation and System Requirements

Please see the Hardware and Software sections for information regarding preinstallation system requirements.

4.2. Platform Installation and Preparation

JLV is being installed in the VAEC using Fargate Containers. The ECR stores and manages the container images.

Docker technology is used to create file images. Docker is a set of "platform as a service" (PAAS) products that use Operating System (OS)-level virtualization to deliver software in packages called containers

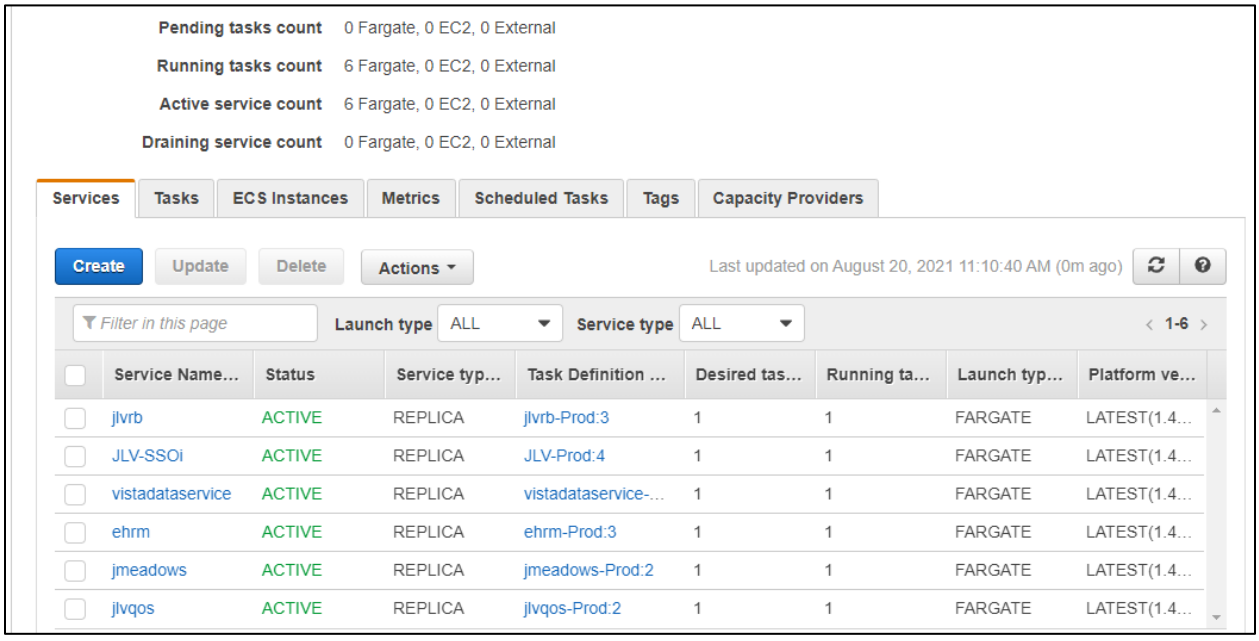
When a Docker file is built, the configuration instructions are executed, and the artifacts are installed into the image. This activity creates the container image.

Docker images are packaged up and sent to ECR. When changes are made to individual services, a new Docker image is created, and the new image is sent to ECR, where a new container is created to replace the outdated container.

Table 8: Implementation Plan Summary

Considerations	Associated Details	
What systems are affected?	**Component:**	**Deployed to:**
	JLV Web Application	VAEC Environment
	JLV Report Builder	VAEC Environment
	jMeadows Data Service	VAEC Environment
	EHRM Service	VAEC Environment
	VDS	VAEC Environment
	JLV QoS	VAEC Environment
Who is impacted by the change?	JLV users	
What is the estimated timeframe for restoring service?	Estimated down time is 60 minutes for deployment and testing.	
What pre-implementation work is required?	Build Fargate Docker images using AWS CodeBuild. AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy.	

Figure 5: Screenshot of JLV Cluster



JLV Fargate Cluster is a regional grouping of containers which together form the JLV application.

- There is a setting in the cluster which describes the number of containers that are needed at any one time.

    - Anytime one of the containers crashes, this service will automatically create a new container to ensure the desired number of containers is maintained.
- Each container is described by the task definition associated with each specific container.
- The JLV Task Definition is stored in ECS.
- The ECR stores and manage container images.

4.3. Download and Extract Files

The application code is pulled into the Docker file from GitHub and built during the container build process. Configuration files and certificates are pulled into the Docker file from a Nexus artifact repository located on a server in the AWS JLV domain.

4.4. Database (DB) Creation

The JLV DB is being migrated and restored in Amazon RDS. It is a SQL Server Enterprise Edition DB, used to store user profile information, audit records, and medical standard translation and mapping reference tables.

System design specifications and diagrams can be found in the VA JLV Product Repository on GitHub. See Purpose for the link to the repository.

4.5. Installation Scripts

This does not apply to JLV. There are only Docker files which are deployment templates for creating images. Docker will install any software, configurations, and artifacts needed.

4.6. Cron Scripts

Cron scripts are Linux-based events. There are currently no cron scripts run for JLV.

The singular, regularly scheduled event is the hourly server health check, a Windows-based event.

4.7. Access Requirements and Skills Needed for Installation

AWS Console or Command Line Interface (CLI) access is required for installation activities. JLV System Engineers have been granted access, and they are designated to access the jump servers for deployment, maintenance, and backout activities. (A jump server, jump host or jump box is a system on a network used to access and manage devices in a separate security zone.) This document assumes the installer has knowledge and experience with the AWS cloud platform, Docker containers, MS SQL Server, Linux, Apache, and Tomcat, in addition to a general understanding of the web-based applications and familiarity with networking and basic troubleshooting, such as Telnet and ping.

4.8. Installation Procedures

The subsections below detail pre-installation, and installation procedures. See Purpose for the link to the repository.

Note: Estimated down time is 60 minutes for deployment and testing.

4.8.1. Preinstallation Procedures

Record the JLV software version number to be installed (for reference), as well as the software version number of the previous installation

If the database changes are needed, follow the steps below in section 4.8.2.7.

Once the preinstallation activities are complete, the installation of JLV system components begins in the primary environment.

4.8.2. Installation in ECS Fargate Cluster Environments

Complete these installation steps and validate the installation according to the steps in Installation Verification Procedures. See the Deployment workflow for the detailed process.

4.8.2.1. Update JLVQoS Package.

1. Login to AWS CodeBuild in the Console.
2. Navigate to Build projects and select “jlvqos-prod”
3. Select Start build with overrides"
4. Go to the Environment variables override section and expand the details.
5. Update the "IMAGE_TAG" variable with the proper tag and verify other environment variables.
6. Select "Start build".
7. Go to AWS ECS in the Console and select “Clusters”.
8. Select the JLV-Prod-A cluster
9. Select the JLV-QoS task
10. Stop the JLV-QoS task – AWS will automatically start a new task with the latest version

4.8.2.2. Update JLVRB Package

1. Login to AWS CodeBuild in the Console.
2. Navigate to Build projects and select “reportbuilder-prod”
3. Select Start build with overrides"
4. Go to the Environment variables override section and expand the details.
5. Update the "IMAGE_TAG" variable with the proper tag and verify other environment variables.
6. Select "Start build".
7. Go to AWS ECS in the Console and select “Clusters”.
8. Select the JLV-Prod- A cluster
9. Select the JLVRB task
10. Stop the JLVRB task- AWS will automatically start a new task with the latest version.

4.8.2.3. Update JLV Package

1. Login to AWS CodeBuild in the Console.
2. Navigate to Build projects and select “jlv-web-prod”
3. Select Start build with overrides"
4. Go to the Environment variables override section and expand the details.

5. Update the "IMAGE_TAG" variable with the proper tag and verify other environment variables.
6. Select "Start build".
7. Go to AWS ECS in the Console and select “Clusters”.
8. Select the JLV-Prod- A cluster
9. Select the JLV task
10. Stop the JLV task – AWS will automatically start a new task with the latest version

4.8.2.4. Update VistADataService Package

1. Login to AWS CodeBuild in the Console.
2. Navigate to Build projects and select “vds-prod”
3. Select Start build with overrides"
4. Go to the Environment variables override section and expand the details.
5. Update the "IMAGE_TAG" variable with the proper tag and verify other environment variables.
6. Select "Start build".
7. Go to AWS ECS in the Console and select “Clusters”.
8. Select the JLV-Prod- A cluster
9. Select the VistaDataService task
10. Stop the VistaDataServicee task – AWS will automatically start a new task with the latest version

4.8.2.5. Update jMeadows Package

1. Login to AWS CodeBuild in the Console.
2. Navigate to Build projects and select “jmeadows-prod”
3. Select Start build with overrides"
4. Go to the Environment variables override section and expand the details.
5. Update the "IMAGE_TAG" variable with the proper tag and verify other environment variables.
6. Select "Start build".
7. Go to AWS ECS in the Console and select “Clusters”.
8. Select the JLV-Prod- A cluster
9. Select the jMeadows task
10. Stop the jMeadows task – AWS will automatically start a new task with the latest version

4.8.2.6. Install Electronic Health Record Modernization (EHRM) Service

1. Upload EHRM artifact to Nexus repository
2. Update EHRM Dockerfile to point to new artifact
3. Login to AWS CodeBuild in the Console.
4. Navigate to Build projects and select “ehrm-prod”
5. Select Start build with overrides"
6. Go to the Environment variables override section and expand the details.
7. Update the "IMAGE_TAG" variable with the proper tag and verify other environment variables.
8. Select "Start build".
9. Go to AWS ECS in the Console and select “Clusters”.
10. Select the “JLV-EHRM” task

11. Stop the JLV-EHRM task – it will automatically start a new task with the latest version

4.8.2.7. Steps for Database Updates

1. Remote desktop into the Windows jumpbox
2. Open Microsoft SQL Server Management Studio (SSMS)
3. Connect to *RDS instance*
4. Open the SQL script *<filename>.sql*, provided with the JLV source code package submission
5. Execute the SQL script *<filename>.sql*

4.9. Installation Verification Procedures

After completing the installation processes detailed in Installation Procedures the Operations Team performs a manual smoke test. Use the steps below to test each module as an end user to validate the installation, deployment, and functionality of all JLV applications and services.

1. Validate that JLV is running
    a. Access the JLV application web page using the following URL: REDACTED
2. Validate that VA Personal Identification Verification (PIV) and Personal Identification Number (PIN) are accepted by SSOi
3. Validate that the system status appears on the JLV **Login** page
    a. **Expected Result:** The system status should show a circular, green icon with a white checkmark
4. Validate the ability to log in with VA credentials
5. Validate that VA data displays within the JLV widgets, using test patients CHDR 1 and CHDR 2
6. Validate that FEHR data displays within the JLV widgets
7. Validate that community partner data displays within the JLV widgets
8. Validate that VA terminology mapping occurs
    a. **Expected Result:** VA terminology is properly mapped in the JLV widgets
9. Validate that DOD terminology mapping occurs
    a. **Expected Result:** DOD terminology is properly mapped in the JLV widgets
10. Validate that QoS is running by verifying that QoS is writing updates to the DB in the QoS_LOGS table
    a. Run the following command in SSMS on the active MSSQL server: select top 100 * from jlv.dbo.QOS_LOGS and order by date desc
        i. **Expected Result:** The select top 100* results will be displayed and indicate a time stamp of within 5 minutes of the time the query was run
        ii. If the top rows do not show, double check the installation steps
11. Validate that Report Builder is running
    a. **Expected Result:** The user can add items to the Report Builder and generate a printable Portable Document Format (PDF) file
12. Once all verification steps are complete, JLV users are rerouted back to the primary Production operating environment

4.10. System Configuration

Table 5 describes the server configurations for JLV Enterprise Production infrastructure hosted at the VAEC

4.11. DB Tuning

JLV Engineering/Development ensure DB indexing for each release. The DB schema and the performance-based jobs are validated by JLV Database Administrators (DBAs) for the deployment of each release.

5. Backout Procedures

A backout is performed before a rollback. The backout procedures remove the newly installed components if the JLV deployment did not pass the installation verification procedures. Both backout and rollback are performed consecutively for each JLV component to return to the last known good operational state of the software and platform settings.

5.1. Backout Strategy

The backout strategy is to uninstall the currently deployed JLV system components and restore the previously deployed version of JLV using the instructions listed in Rollback Procedures.

5.2. Backout Considerations

The following subsections detail the considerations for backing out of the current installation of JLV.

5.2.1. Load Testing

Load testing is currently being coordinated with the VA Enterprise Testing Service (ETS) team.

5.2.2. User Acceptance Testing (UAT)

UAT results were not available at the time of this writing. When all testing cycles (including UAT) are complete, the data is made available in the VA JLV Product Repository in GitHub. See Purpose for the link to the repository.

5.3. Backout Criterion

The criterion for backing out of the current installation is that JLV does not operate as intended when tested by VA and partner testers and the JLV Support team.

5.4. Backout Risks

The risks for executing the backout are minimal because a backout is performed during planned downtime when users are not accessing the system. Once the restored system is online and validated, user access continues.

If a backout is initiated later in the deployment window, restoration time may exceed the downtime planned for deployment. This risk is mitigated by scheduling deployments for weekends and other times when expected usage levels are low.

5.5. Authority for Backout

If a backout is necessary, approval for the backout comes from the VA PgM REDACTED or VA PM REDACTED

5.6. Backout Procedures

Because backout and rollback are performed consecutively, the backout and rollback procedures are combined in Rollback Procedures.

5.7. Backout Verification Procedures

See Installation Verification Procedures.

6. Rollback Procedures

A rollback is performed after a backout. The rollback procedures restore the previously deployed version of JLV.

6.1. Rollback Considerations

The consideration for performing a rollback is that the JLV application does not operate as intended when tested by the JLV Support team.

6.2. Rollback Criterion

The criterion for performing a rollback is that the JLV application does not operate as intended when tested by the JLV Support team.

6.3. Rollback Risks

The risks for executing a rollback are minimal because the system will revert to the last working image/configuration.

6.4. Authority for Rollback

If a rollback is necessary, approval for the rollback comes from the VA PgM REDACTED or VA PM REDACTED

6.5. Rollback Procedures

AWS ECR will maintain versions of previous releases. After the initial rollout of the JLV Cloud instance, a rollback would involve restarting a previous version of the JLV application.

If it is determined that a rollback is necessary, task definitions with the previous image version will be deployed to ECS.

6.6. Rollback Verification Procedures

After completing the rollback procedures, perform the validation steps in Installation Verification Procedures.

A. Acronyms and Abbreviations

Table 9 lists the acronyms and abbreviations are used throughout this document.

Table 9: Acronyms and Abbreviations

Acronym	Definition
AD	Active Directory
AITC	Austin Information Technology Center
ALB	Application Load Balancer
API	Application Programming Interface
ATO	Authority to Operate
AWS	Amazon Web Services
BHIE	Bidirectional Health Information Exchange
CA	Computer Associates
CAC	Common Access Card
CD2	Critical Decision Point 2
CDR	Clinical Data Repository
CM	Change Management
CHDR	Clinical/Health Data Repository
CHG	Change Order
COR	Contracting Officer's Representative
CVIX	Central VistA Imaging Exchange
DAS	Data Access Service
DB	Database
DBA	Database Administrator
DES	Data Exchange Service
DIBRG	Deployment, Installation, Backout, and Rollback Guide
DRP	Dynamic Routing Protocol
DoD	Department of Defense
ECR	Elastic Container Registry
ECS	Elastic Container Service
EC2	Elastic Compute Cloud
EHR	Electronic Health Record
EHRM	Electronic Health Record Modernization
eHX	eHealth Exchange
EP	Elevated Privilege
ePAS	Electronic Permission Access System
EPMO	Enterprise Program Management Office
FEHR	Federal Electronic Health Record
FHIR	Fast Healthcare Interoperability Resources
GB	Gigabyte
GTM	Global Traffic Manager
GUI	Graphical User Interface
HAIMS	Healthcare Artifact and Image Management Solution

Acronym	Definition
HIE	Health Information Exchange
HRG	Hawaii Resources Group
HTTPS	Hypertext Transfer Protocol Secure
IAM	Identity and Access Management
IP	Internet Protocol
IO	Infrastructure Operations
IT	Information Technology
JDBC	Java Database Connectivity
JLV	Joint Longitudinal Viewer
LDAP	Lightweight Directory Access Protocol
MHS	Military Health System
MS	Microsoft
MPI	Master Person Index
NCA	National Cemetery Administration
OEHRM	Office of Electronic Health Record Modernization
OIT	Office of Information and Technology
OS	Operating System
PAAS	Platform As A Service
PCMM	Patient Centered Management Module
PDWS	Patient Discovery Web Service
PDF	Portable Document Format
PITC	Philadelphia Information Technology Center
PIN	Personal Identification Number
PIV	Personal Identification Verification
PM	Program Manager or Project Manager
POC	Point of Contact
POM	Production Operations Manual
QoS	Quality of Service
RAM	Random Access Memory
REST	Representational State Transfer
RPC	Remote Procedure Calls
RRR	Release Readiness Report
SOAP	Simple Object Access Protocol
SMS	Systems Made Simple
SNOW	Service Now
SSMS	SQL Server Management Studio
SSOi	Single Sign on Internal
SQL	Structured Query Language
TCP	Transmission Control Protocol
TMDS	Theater Medical Data Store
UAT	User Acceptance Testing
URL	Universal Resource Locator

Acronym	Definition
U.S.	United States
VA	Department of Veterans Affairs
VAEC	VA Enterprise Cloud
VBA	Veterans Benefits Administration
VDS	VistA Data Service
VHA	Veterans Health Administration
VIP	Veteran-Focused Integration Process
VistA	Veterans Information Systems and Technology Architecture
VLER	Virtual Lifetime Electronic Record
VM	Virtual Machine
YAML	Yet Another Markup Language